ISCR

Institute for Scientific Computing Research

Visiting and Collaborating Professors' Project Abstracts

CASC Center for Applied Scientific Computing

Applying ATLAS Technology to ASCI-Related Applications

Jack Dongarra University of Tennessee

Abstract

Today's microprocessors have peak execution rates of gigaflop/s. However, straightforward implementation in Fortran or C of computations based on simple loops rarely results in such high performance. To realize such peak rates of execution for even the simplest of operations has required tedious, hand-coded, programming efforts.

In general, the existing BLAS have proven to be very effective in assisting portable, efficient software for sequential, vector and shared-memory high-performance computers. Hand-optimized BLAS are expensive, however, tedious to produce for any particular architecture, and in general will only be created when there is a large enough market, which is not true for all platforms. The process of generating an optimized set of BLAS for a new architecture or a slightly different machine version can be a time-consuming process. The programmer must understand the architecture, how the memory hierarchy can be used to provide data in an optimum fashion, how the functional units and registers can be manipulated to generate the correct operands at the correct time, and how best to use the compiler optimization. Care must be taken to optimize the operations to account for many parameters such as blocking factors, loop unrolling depths, software pipelining strategies, loop ordering, register allocations, and instruction scheduling.

We have developed a general methodology for the generation of the efficient linear algebra kernels. In our approach, we have isolated the machine-specific features of the operation to several routines, all of which deal with performing an optimized on-chip, cache contained, (i.e., in Level 1 (L1) cache) matrix multiply. This section of code is automatically created by a code generator that uses timings to determine the correct blocking and loop-unrolling factors to perform an optimized on-chip multiply. The user may directly supply the code generator with as much detail as desired (e.g., the user may explicitly indicate the L1 cache size, the blocking factor(s) to try, etc). If such details are not provided, the generator will determine appropriate settings via timings. Our approach, called Automatically Tuned Linear Algebra Software (ATLAS), has been able to match or exceed the performance of the vendor-supplied version of matrix multiply in almost every case.

This package has been designed to encode and efficiently take advantage of hardware information. Currently, our automated optimization techniques are applied to generate highly efficient implementations of dense linear algebra kernels such as the BLAS, since these subprograms are essential computational building blocks of many scientific computing codes, as well as heavily used general purpose software libraries such as LAPACK or PETSc. Our research is well advanced, and has demonstrated the

Applying ATLAS Technology to ASCI-Related Applications (continued)

capability of automatically generating, for a wide range of computers, a number of highly efficient kernels achieving performance comparable to, and often better than, hand-optimized codes (often written in assembler) tailored specifically for a particular architecture. These encouraging results are in addition immediately and directly re-usable and applicable to software designed and written for distributed-memory concurrent computers, since these optimized kernels are also heavily used locally by each processor in many scientific computing application areas.

Determining and characterizing the effectiveness of run-time optimization techniques is particularly relevant to modern sparse linear algebra software libraries that tend to hide from the user the internal storage format. For instance, a certain storage structure may dictate a particular algorithmic approach. Once a general scheme of access has been found based on given a storage structure, one promising idea involves analyzing the sparsity pattern of the matrix operand to find places where the efficient static optimization techniques may be re-used. There are numerous ways in which this can be done, and it is almost certain that there will be no provably best way. In this case, it will also be necessary to search the space of available options during run-time.

We are experimenting with a variety of techniques for optimizing sparse matrix vector multiplication to take instruction sets, functional units, and memory hierarchies into account. Sparse matrix–vector multiplication is of course the inner loop in any iterative solver, even multigrid, as it includes all the interpolation, restriction, and smoothing operations. The structural properties of the application lead to sparse matrices that feature a sufficiently regular pattern, so that the automatic optimization techniques already integrated in ATLAS can be successfully re-used and applied to generate the appropriate basic sparse linear algebra kernels needed in many applications. Our plan for achieving the necessary and exceptionally high degree of portability and optimization leverages the experience of our team in developing ATLAS technology.

Developing a Tuned Version of ScaLAPACK's Linear Equation Solver

Jack Dongarra University of Tennessee

Abstract

The LINPACK benchmark has been used as a yardstick in measuring the performance of the Top500 installed high-end computers. This benchmark was chosen because it is widely used and performance numbers are available for almost all relevant systems. The approach used in the LINPACK benchmark is to solve a dense system of linear equations. For the Top500, the benchmark allows the user to scale the size of the problem, and to optimize the software in order to achieve the best performance for a given machine. This performance does not reflect the overall performance of a given system, as no single number ever can. It does, however, reflect the performance of a dedicated system for solving a dense system of linear equations. Since the problem is very regular, the performance achieved is quite high, and the performance numbers give a good check of peak performance of a system.

By measuring the actual performance for different problem sizes n, a user can get not only the maximal achieved performance Rmax for the problem size Nmax but also the problem size $N_{1/2}$ where half of the performance Rmax is achieved. These numbers together with the theoretical peak performance Rpeak are the numbers given in the Top500. In an attempt to obtain uniformity across all computers in performance reporting, the algorithm used in solving the system of equations must confirm to the standard operation count for LU factorization with partial pivoting. In particular, the operation count for the algorithm must be $2/3n^3 + 0(n^2)$ floating point operations.

Drawing upon our years of experience with ScaLAPACK and BLACS development, we have developed a version of the benchmark based on the hardware of the ASCI Blue–Pacific. Collaborating with Andrew Cleary at LLNL, we have achieved a performance of 2.144 teraflop/s. One major obstacle in achieving maximal performance with the present ASCI Blue setup is that it involves using multiple machines for one run. This problem has been shown to cause a loss of efficiency of approximately 20% on the current benchmark code on ASCI Blue.

Research on Parallel Adaptive Finite Element Methods

Michael J. Holst University of California, San Diego

Abstract

We are developing and implementing parallel algorithms for the adaptive solution of systems of partial differential equations (PDEs) using the finite element method (FEM). PDEs lie at the heart of many problems in scientific computing, as many physical laws are most conveniently expressed through them. Many PDEs are derived from variational formulations of physical problems. These equations are so large that only the most efficient algorithms can be employed, and only those that are scalable to massively parallel computers have any chance of success. Moreover, since it is often necessary to model complicated geometries, many PDEs are discretized on general unstructured grids and the solver must perform efficiently without structured geometric grid information.

We have focused on several fundamental issues arising in the parallel adaptive solution of linear and nonlinear elliptic PDEs. Adaptive meshing algorithms are critical to the successful solution of many classes of PDES. One challenging problem in this area is to incorporate such adaptive algorithms into a parallel-computing environment, since the final mesh (and hence the load balance) is usually not known *a priori*. Our UCSD group has developed a procedure based on globally defined grids, each of which is fully refined in an exclusive local region, which leads to efficient parallel adaptive methods, using minimal communication and substantial reuse of existing quality sequential software.

Numerical Methods for Partial Differential Equations in Large-Scale Scientific Computations

Raytcho D. Lazarov Texas A&M University

Abstract

Construction, analysis and numerical testing of efficient discretization techniques for solving elliptic partial differential equations that allow for parallel implementation are the foci of our research. This ultimately imposes domain decomposition-type algorithms, in which each subdomain is uniquely assigned to a processor. The case when the meshes do not align on the interfaces between subdomains is considered. This situation occurs when either coarsening or refinement is done independently and in parallel on the subdomains, generating grids that do not match along the interfaces between the subdomains. Mortar finite element techniques to glue the solutions across the subdomain boundaries have been employed. We also study a least-squares stabilization technique for solving advection–diffusion problems and problems of linear elasticity using the "minus one" norm inner product.

Work on the *hypre* Framework and Molecular Simulations on Massively Parallel Processors

Calvin Ribbens Virginia Polytechnic Institute and State University

Abstract

Our work has two separate foci: scalable linear solvers and computational materials science.

We are assisting in the design, development, implementation, and testing of the *hypre* framework and library. Our contribution shows how a variety of domain decomposition preconditioners fit into the framework. Also, the ability of the larger Equation Solver Interface (ESI) standard to incorporate these preconditioned solvers will be considered. We are also studying the design and performance of output coefficient access with regard to the CoefficientAccess class of the current *hypre* design. The output method on objects of this class allows preconditioners to read coefficients of a matrix, e.g., as is needed by ILU. The tradeoffs between efficiency ("How quickly can I get the data and how well do I use the memory hierarchy?") and expressiveness and interoperability ("How big is the set of preconditioners that can be easily expressed?") need more study. We are also studying the design of the ESI and *hypre* in terms of support for two-level, inner-outer, preconditioned solvers. We are investigating new algorithms of this type that combine domain decomposition approaches with multigrid in an effort to improve the applicability of multigrid without sacrificing scalability.

We have been collaborating for more than two years with Diana Farkas of the Materials Science and Engineering Department at Virginia Tech on work focused on molecular simulations on massively parallel machines. The PI is working with LLNL's Patrice Turchi to incorporate the new real-space Electronic Structure (ES) approach to study the interplay between chemical order and topological disorder in complex bulk amorphous alloys. Specifically, we are modifying the existing 0(N) Tight-Binding Molecular Dynamics (TBMD) codes, already implemented on the T3E at NERSC, to account for the d-electron behavior in materials. This extension has already been done in an $0(N^3)$ serial version of the codes. The new MD codes will be implemented on ASCI Blue–Pacific. We will then extend the TBMD codes to a full spd-atomic orbital basis set and to multi-species (the extension has already been done in an $0(N^3)$ serial version of the codes). All of this is done with a view toward parallelizing the ES and Monte Carlo (MC) parts of the scheme on the ASCI Blue machine.

Once parallel versions of the codes have been developed, the goal is to perform simulations on large systems (1,000-10,000 atoms) using an extended version of the 0(N) TB-MD parallel codes. The current codes require on the order of tens of hours (on a 32-node T3E) to perform a structural minimization of a simple system containing about 1,000 atoms. Several iterations of this minimization are needed to study different structures with a specific thermal history. To store the wavefunctions and atomic coordinates at intermediate steps (for later analysis), and also the final configuration, 50 MB of disk space and 50 GB of storage are required. A single configuration on a 32 PE system (with 1,000 atoms) is typically of the order of 1 MB. Note that the TB-MD codes have already been parallelized on a Cray environment. The other codes (ES and MC) require parallelization to make the entire loop efficient. The real-space formalism will lead to a full implementation of the codes on MPP machines.

Scalable Linear Solvers for Partial Differential Equations

Jinchao Xu Pennsylvania State University

Abstract

Our work concerns scalable linear solvers with special emphasis on multigrid/multilevel methods. Multigrid methods are deemed to be one of the most powerful methods for solving large-scale algebraic systems arising from the discretization of partial differential equations. Perhaps partially because of their mathematical complexity and problem-dependent tuning and performance, multigrid methods are underutilized in ASCI applications. One particular area of fruitful collaboration is the development of the so-called algebraic multigrid methods, in which our experience ranges from theoretical analysis to code development.

Our particular expertise is efficient methods for convection-dominated convection–diffusion problems, including new monotone finite element schemes, ordering algorithms for effective Gauss–Seidel iteration, and efficient multigrid algorithms for unstructured grids.

Another area of collaboration is adaptive grid techniques in finite element discretizations. Whereas adaptive finite element grid refinement has mostly been done in an "unstructured" fashion, "locally structured" refinement technique appears to have great potential for efficient finite element implementation on high-performance computers.